PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Application Number:	10/820295
Confirmation Number:	7444
Filing Date:	April 8, 2004
Applicants:	Brian CONNELL, Greg WILBUR, John AFAGANIS, Ian MacFARLANE and Lori-Ann McGRATH
Title:	INTEGRATED MODELING ENVIRONMENT
Examiner:	Jason D. MITCHELL
Group Art Unit:	2193
Attorney Docket No:	22493-27U (16666ROUS01U)

Mail Stop Appeal Brief - Patents
Commissioner For Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPEAL BRIEF

Sir:

This Appeal Brief is submitted in support of the Notice of Appeal filed October 12, 2009, and in response to the Final Office Action dated July 21, 2009, wherein Appellant appeals from the Examiner's rejection of Claims 1-29.

TABLE OF CONTENTS

I. Real Party in Interest

The real party in interest is Nortel Networks Limited, which is the assignee of the subject application by virtue of assignment recorded on Reel/Frame 015071/0318 on August 18, 2004.

II. Related Appeals and Interferences

None.

III. Status of Claims

Claims 1-29 are pending in this Application. Claims 1-29 have been finally rejected, and it is from the final rejection of Claims 1-29 that this Appeal is taken.

IV. Status of Amendments

An Amendment after filing the Notice of Appeal was filed on December 14, 2009 to correct a typographical error. The amendment addressed the rejection under 35 U.S.C. § 112, second paragraph and was filed solely to reduce the issues on Appeal. Specifically, Claim 21 has been amended to depend from dependent Claim 20 rather than directly from independent Claim 19. It is believed this amendment obviates the rejection under 35 U.S.C. § 112.

V. Summary of Claimed Subject Matter

The present invention, as recited in independent Claims 1, 13 and 19, is directed toward a method, system, and computer program product for modeling network elements as described throughout the Specification including in the Summary of the Invention, page 1, line 15 through page 6, line 10.

With respect to independent Claim 1, the method includes modeling network element commands, events and run-time system data into a data model using a first modeling language, where the data model comprises first data (See FIG. 2, and page 7, line 18 through page 8, lines 9). The method translates the first data represented in the first modeling language to second data represented in a second modeling language (See page 9, lines 13-19). The method stores the second data in the second modeling language in a global data model repository (See FIG. 2 and page 7, line 22 through page 8 line 5), and automatically generates code to support an external management interface based on the stored second data in the global repository, where the external management interface communicates with the stored second data (page 8, line 7 through page 9, line 12).

Independent Claim 13 recites a modeling system that includes a memory comprising a global repository (See FIG. 2), a processor electrically coupled to the memory (See FIG. 2, and page, 7, line 22 through page 8, line 9), a first interface to a plurality of network elements where the first interface is in communication with the global repository (See FIGS. 1 and 2, and page 6, line 22 through page 7, line 17, and page 8, lines 6-8), and a second interface to an external interface where the second interface is in communication with the global repository (See FIG. 2. and page 8, lines 6-8). The processor is configured to model network element commands, events and run-time system data into a data model using a first modeling language where the data model comprises first data (See page 9, lines 13-19), translate the first data represented in the first modeling language to second data represented in a second modeling language (See page 9, lines 13-19), store the second data in the second modeling language in the global data model repository (See FIG. 2 and page 7, line 22 through page 8 line 5), and automatically generate code to support an external management interface code development based on the stored second

data in the global repository, where the external management interface communicates with the stored second data (See page 8, line 7 through page 9, line 12).

Independent Claim 19 recites a computer program product tangibly embodied in a computer storage medium for executing instructions on a processor. The computer program product is operable to cause a machine to model network element commands, events and run-time system data into a data model using a first modeling language, the data model comprising first data (See FIG. 2, and page 7, line 18 through page 8, lines 9), translate the first data represented in the first modeling language to second data represented in a second modeling language (See page 9, lines 13-19), store the second data in the second modeling language in a memory comprising a global data model repository (See FIG. 2 and page 7, line 22 through page 8 line 5), and automatically generate code to support an external management interface code development based on the stored second data in the global repository, the external management interface communicating with the stored second data (See page 8, line 7 through page 9, line 12).

VI. Grounds of Rejection to be Reviewed on Appeal

Claims 1-4, 8-10, 12-16, 18-22, 24, 26-27 and 29 were rejected under 35 U.S.C. §102(b) as being unpatentable in view of U.S. Publication No. 2002/0091809 to Menzies et al. ("Menzies").

Claims 5-7, 11, 17, 23, 25 and 28 were rejected under 35 U.S.C. §103(a) as being unpatentable over Menzies in view of U.S. Publication No. 2003/0046370 to Courtney ("Courtney").

VII. Argument

The Rejection of Claims 1-4, 8-10, 12-16, 18-22, 24, 26-27 and 29 under 35 U.S.C. §102(b)

Claims 1-4, 8-10, 12-16, 18-22, 24, 26-27 and 29 were rejected under 35 U.S.C. §102(b) as being unpatentable in view of U.S. Publication No. 2002/0091809 to Menzies et al. ("Menzies"). For convenience of the Honorable Board in addressing the rejections, dependent Claims 2-4, 8-10, and 12 stand or fall with independent Claim 1, dependent Claims 14-16 and 18 stand or fall with independent Claim 13, and dependent Claims 20-22, 24, 26-27 and 29 stand or fall with independent Claim 19.

A. The Examiner fails to cite a reference disclosing each and every element of Applicants' Claimed Invention

Independent Claims 1, 13 and 19 each recite the features of "modeling network element commands, events and run-time system data into a data model using a first modeling language, the data model comprising first data; translating the first data represented in the first modeling language to second data represented in a second modeling language … and ***automatically generating code*** to support an external management interface based on the stored ***second*** data in the global repository, the external management interface communicating with the stored ***second*** data." These features are not taught, disclosed, or suggested by Menzies.

The data model in the present invention and as recited in Claims 1, 13 and 19 is more than just a configuration file, as it includes commands, events (such as alarms, statistics, counters, etc.) and run-time system data. For example, the network element gathers network element data at run-time from multiple components or software processes that comprise the network element. This operational run-time data provides network operators the ability to keep,

for instance, a router's statistics and accounting data, as well as information needed to manage the router. The network element, e.g. the router, is managed through "commands, events and run-time system data" to provide, among others, maintenance and billing information. This arrangement is not taught or suggested by Menzies.

Menzies discloses a method and system for providing mapping between an SNMP MIB module scheme and a Common Information Model (CIM) scheme. The mapping process enumerates the MIB objects and then maps the objects into CIM Managed Object Format (MOF) classes using defined mapping tables. A correlation mechanism is provided to determine in real time which MIBs (or portions thereof) and corresponding CIM classes a network device supports.

Initially, Menzies fails to disclose "translating the first data represented in the first modeling language to second data represented in a second modeling language". The Examiner cites paragraph [0064] in Menzies for allegedly disclosing this feature. However, Menzies merely maps management information from one schema (MIB) to another (CIM). FIG. 5 of Menzies shows how MIB information is mapped to CIM classes. In the CIM installation, each MIB is extracted from an SNMB MIB repository (SMIR) and translated by an SMI (Structure of Management Information) compiler into a CIM object class, which is provided to the CIM object manager CIMOM and stored in the CIM repository. To accomplish this, the SMI compiler references a number of mapping tables to facilitate the translation. The CIMOM combines the dynamic information with the static MIB information now in the form of CIM classes to produce a desired instance output. Menzies does not "model network element commands, events and run-time system data into a data model using a first modeling language" or "translate the first

data represented in the first modeling language to second data represented in a second modeling language". Claims 1, 13 and 19 are believed patentable over Menzies for at least this reason.

The Examiner also relies on Menzies's ability to "transparently translate the user's high-level query into a series of simple retrievals and then perform the query internally on behalf of the user" (paragraph [0060]) as somehow equating with Applicants' "automatically generating code to support an external management interface based on the stored second data in the global repository, the external management interface communicating with the stored second data". However, Menzies is not "automatically generating code to support an external management interface based on the stored second data in the global repository". Instead, Menzies is merely taking a user's high-level query, such as an SQL query, and translating the query into a series of simpler queries, "giving the impression that the provider directly supported the query". (Menzies, paragraph [0060]). Menzies does not automatically generate code based on stored second data in a global repository. Claims 1, 13 and 19 are believed patentable over Menzies for at least this additional reason.

In sum, despite the Examiner's statements in the Final Office Action, Menzies, the sole reference cited in the 35 U.S.C. §102 rejection, fails to disclose every element recited in Applicants' independent Claims 1, 13, and 19. Specifically, Menzies fails to disclose at least "modeling network element commands, events and run-time system data into a data model using a first modeling language, the data model comprising first data", "translating the first data represented in the first modeling language to second data represented in a second modeling language" and "automatically generating code to support an external management interface based on the stored second data in the global repository, the external management interface communicating with the stored second data."

Concluding, the Examiner has failed to cite a reference that discloses each and every element of Applicants' claims as required for a rejection under 35 U.S.C. §102. Accordingly, the Examiner's rejection with respect to Claims 1-4, 8-10, 12-16, 18-22, 24, 26-27 and 29 should be reversed.

The Rejection of Claims 5-7, 11, 17, 23, 25 and 28 under 35 U.S.C. §103(a)

Claims 5-7, 11, 17, 23, 25 and 28 were rejected under 35 U.S.C. §103(a) as being unpatentable over Menzies in view of U.S. Publication No. 2003/0046370 to Courtney ("Courtney").

A. The Examiner fails to cite a reference or combination of references disclosing each and every element of Applicants' Claimed Invention

As an initial matter, Applicants note that dependent 5-7, 11, 17, 23, 25 and 28 are believed patentable at least by virtue of their dependency on one or another of their base independent Claims 1, 13 and 17. The patentability of independent Claims 1, 13 and 17 with respect to Menzies are discussed above.

As discussed above, Menzies fails to disclose a number of claimed features including at least "modeling network element commands, events and run-time system data into a data model using a first modeling language, the data model comprising first data", "translating the first data represented in the first modeling language to second data represented in a second modeling language" and "automatically generating code to support an external management interface based on the stored second data in the global repository, the external management interface communicating with the stored second data." Apparently, Courtney was cited for its disclosure of XML as a modeling language (Final Office Action, page 7), generating code to implement a

command line interface (CLI) (Final Office Action, page 8) and generating code to assist in the implementation of an Application Program Interface (Final Office Action, page 9).

Courtney fails to disclose the features of Applicants' claims that are not taught or suggested by Menzies. Courtney merely deals with configuring network equipment by simply converting a router's commands from a native representation into a standard format. Courtney does not teach an interface that interacts and communicates with a stored data model, because Courtney does not store a data model at all. Courtney says nothing about a data model that models a network element's commands, events, and run-time system data. Courtney merely creates commands in a standard format to interface with the routers themselves, not with a data model representation of the router. In fact, nothing in Courtney teaches the concept of a complete data model representation of a network element, where the data model itself is used to automatically generate code for an interface that communicates and interacts with the stored data model.

Thus, even if the features of Menzies and Courtney were somehow combined, the resulting system would still not include each of Applicants' claimed features as recited in Claims 5-7, 11, 17, 23, 25 and 28. The combined system still does not disclose or teach the claimed steps of "modeling network element commands, events and run-time system data into a data model using a first modeling language, the data model comprising first data", "translating the first data represented in the first modeling language to second data represented in a second modeling language" and "automatically generating code to support an external management interface based on the stored second data in the global repository, the external management interface communicating with the stored second data."

Thus, the Examiner has failed to cite a combination of references disclosing each and

every element of Applicants' claims as required for a *prima facie* case of obviousness. Accordingly, the Examiner's rejection with respect to Claims 5-7, 11, 17, 23, 25 and 28 should be reversed.

VIII. Conclusion

For the reasons provided above as well as provided in the record, the claim rejections are believed to be improper and a result of clear error by the Examiner. Accordingly, pending Claims 1-29 are believed to be in condition for allowance, and a reversal of the Examiner's rejections is respectfully requested.

The Commissioner is hereby authorized to credit overpayments or charge payment of any additional fees associated with this communication to Deposit Account No: 141315.

Respectfully submitted,

Date: December 17, 2009

By: /Alan M. Weisberg/

Alan M. Weisberg
Reg. No: 43,982
Attorney for Applicants
Christopher & Weisberg, P.A.
200 East Las Olas Boulevard, Suite 2040
Fort Lauderdale, Florida 33301
Customer No: 31292
Tel: (954) 828-1488
Fax: (954) 828-9122
email: ptomail@cwiplaw.com

176085

APPENDIX A: CLAIMS ON APPEAL

1. A method comprising:

modeling network element commands, events and run-time system data into a data model using a first modeling language, the data model comprising first data;

translating the first data represented in the first modeling language to second data represented in a second modeling language;

storing the second data in the second modeling language in a global data model repository; and

automatically generating code to support an external management interface based on the stored second data in the global repository, the external management interface communicating with the stored second data.

2. The method of claim 1 further comprising automatically generating system documentation based on the stored second data.

3. The method of claim 2 wherein the generated system documentation corresponds to code generated to support an external management interface.

4. The method of claim 1 wherein the first modeling language is structured management information (SMI).

5. The method of claim 1 wherein the second modeling language is extensible markup language (XML).

6. The method of claim 1 wherein automatically generating code for the external interface includes automatically generating code to implement a command line interface (CLI).

7. The method of claim 1 wherein automatically generating code for the external interface includes automatically generating code to implement an Extensible Markup Language interface.

8. The method of claim 1 wherein automatically generating code for the external interface includes automatically generating code to implement a Simple Network Management Protocol interface.

9. The method of claim 1 wherein automatically generating code for the external interface includes automatically generating code to implement a configuration database.

10. The method of claim 1 wherein automatically generating code for the external interface includes automatically generating code to implement Simple Network Management Protocol subagents.

11. The method of claim 1 wherein automatically generating code for the external interface includes automatically generating code to assist in implementation of an Application Program Interface.

12. The method of claim 1 wherein modeling includes modeling the run-time system data from a plurality of sources using at least one of the first modeling language and the second modeling language.

13. A system comprising:

a memory comprising a global repository;

a processor electrically coupled to the memory;

a first interface to a plurality of network elements, the first interface being in communication with the global repository; and

a second interface to an external interface, the second interface being in communication with the global repository, wherein the processor is configured to:

model network element commands, events and run-time system data into a data model using a first modeling language, the data model comprising first data;

translate the first data represented in the first modeling language to second data represented in a second modeling language;

store the second data in the second modeling language in the global data model repository; and

automatically generate code to support an external management interface code development based on the stored second data in the global repository, the external management interface communicating with the stored second data.

14. The system of claim 13 further configured to automatically generate system documentation based on the stored second data.

15. The system of claim 14 wherein the generated system documentation corresponds to a code generated implementation.

16. The method of claim 13 wherein the first modeling language is structured management information (SMI).

17. The method of claim 13 wherein the second modeling language is extensible markup language (XML).

18. The method of claim 13 wherein the global repository is further configured to model operational system data using at least one of the first modeling language and the second modeling language.

19. A computer program product, tangibly embodied in a computer storage medium, for executing instructions on a processor, the computer program product being operable to cause a machine to:

model network element commands, events and run-time system data into a data model using a first modeling language, the data model comprising first data;

translate the first data represented in the first modeling language to second data represented in a second modeling language;

store the second data in the second modeling language in a memory comprising a global data model repository; and

automatically generate code to support an external management interface code development based on the stored second data in the global repository, the external management interface communicating with the stored second data.

20. The computer program product of claim 19 further configured to automatically generate system documentation based on the stored second data.

21. The computer program product of claim 20 wherein the generated system documentation corresponds to the generated code.

22. The computer program product of claim 19 wherein the first modeling language is structured management information (SMI).

23. The computer program product of claim 19 wherein the second modeling language is extensible markup language (XML).

24. The computer program product of claim 19 wherein the global repository is further configured to model operational system data using at least one of the first modeling language and the second modeling language.

25. The computer program product of claim 19 wherein the instructions to cause a machine to automatically generate code for the external interface include instructions to cause a machine to automatically generate code to implement a command line interface (CLI).

26. The computer program product of claim 19 wherein the instructions to cause a machine to automatically generate code for the external interface include instructions to cause a machine to automatically generate code to implement a configuration database.

27. The computer program product of claim 19 wherein the instructions to cause a machine to automatically generate code for the external interface include instructions to cause a machine to automatically generate code to implement SNMP subagents.

28. The computer program product of claim 19 wherein the instructions to cause a machine to automatically generate code for the external interface include instructions to cause a machine to automatically generate code to implement an API.

29. The computer program product of claim 24 wherein instructions to cause a machine to model operational system data include instructions to cause a machine to model operational system data using at least one of the first modeling language and the second modeling language.

APPENDIX B: EVIDENCE APPENDIX

No evidence submitted pursuant to 37 C.F.R. §§ 1.130, 1.131, or 1.132 of this title or of any other evidence entered by the Examiner has been relied upon by Appellant in this Appeal, and thus no evidence is attached hereto.

APPENDIX C: RELATED PROCEEDINGS APPENDIX

Since Appellant is unaware of any related appeals and interferences, no decision rendered by a court or the Board is attached hereto.